1

Tool Development and Support

SPARK Toolset Release Note–Release 7.4

EXM/RN
Issue: 1.5
Status: Definitive
20th December 2006

Originator

SPARK Team

Approver

SPARK Team Line Manager

Copyright

The contents of this manual are the subject of copyright and all rights in it are reserved. The manual may not be copied, in whole or in part, without the written consent of Praxis High Integrity Systems Limited.

The software tools referred to in this manual are the subject of copyright and all rights in them are reserved. The rights in these tools are owned by Praxis High Integrity Systems Limited, and it may not be copied, in whole or in part, without the written consent of this company, except for reasonable back-up purposes. The same proprietary and copyright notices must be affixed to any permitted copies as were affixed to the original. This exception does not allow copies to be made for others, whether or not sold, and none of the material purchased may be sold, given or loaned to another person or organisation. Under law copying includes translating into another language or format.

©1991-2006 Praxis High Integrity Systems Limited, 20 Manvers St, Bath BA1 1PX

Limited Warranty

Praxis High Integrity Systems Limited save as required by law makes no warranty or representation, either express or implied, with respect to this software, its quality, performance, merchantability or fitness for a purpose. As a result, the licence to use this software is sold ‘as is’ and you, the purchaser, are assuming the entire risk as to its quality and performance.

Praxis High Integrity Systems Limited accepts no liability for direct, indirect, special or consequential damages nor any other legal liability whatsoever and howsoever arising resulting from any defect in the software or its documentation, even if advised of the possibility of such damages. In particular Praxis High Integrity Systems Limited accepts no liability for any programs or data stored or processed using Praxis High Integrity Systems Limited products, including the costs of recovering such programs or data.

SPADE is a trademark of Praxis High Integrity Systems Limited

Note: The SPARK programming language is not sponsored by or affiliated with SPARC International Inc. and is not based on SPARC™ architecture.

1

Contents

1 Introduction 6

2 Contact Information 7

3 SPARK language changes 8
3.1 Accept Annotation 8
3.2 Obsolete floating-point attributes 8
3.3 'Always_Valid assertion 8

4 Examiner changes 9
4.1 Index file syntax changes (SEPR 1954) 9
4.2 Improved syntax error messages for index files (SEPR 1952) 9
4.3 Improvements in VC generation (SEPRs 1968, 1979, 1993) 9
4.4 External variables in loop bounds (SEPR 1943) 9
4.5 Improved diagnosis and reporting of syntax errors (SEPR 1967) 9
4.6 Ada 2005 reserved words warning (SEPR 1987) 10
4.7 Ineffective statement in record assignment (SEPR 1988) 10
4.8 Conditional flow errors are now reported as errors (SEPR 1992) 10
4.9 Unconditional flow errors are now reported more clearly (SEPR 1992) 10
4.10 Disable FlexLM message box pop-up on Windows (SEPR 2000) 11
4.11 Examiner table sizes and limits (SEPR 1997) 11
4.12 FDL Modelling of null string literal "" (SEPR 1176) 11
4.13 VC Generation of formal unconstrained array parameters (SEPR 1084) 11
4.14 Optional mangling of FDL reserved words in proof files output (SEPR 2041) 11
4.15 Examiner exit status (SEPR 2046) 12
4.16 System.Bit_Order available in configuration file (SEPR 2002) 12
4.17 XML and HTML output of justifications (SEPR 1980) 12
4.18 Examiner no longer accepts out of range literals in modular expressions (SEPR 2057) 12
4.19 SPARK95_IO interface change (SEPR 2065) 13
4.20 'Size not static for non scalar types (SEPR 2051) 13
4.21 Metafile handling improved (SEPR 2071) 13
4.22 Ada.Real_Time package definition changed (SEPR 2081) 13
4.23 Detection of illegal constrained array subtype declarations (SEPR 2084) 13

5 SPADE Simplifier 15
5.1 User defined proof rules (SEPR 1969 & 1970) 15
5.2 Standardisation of behaviour of proof rules (SEPR 1974) 15
5.3 New /noexpression_reduction option (SEPR 1986) 15
5.4 Earlier use of ground user-defined replacement rules (SEPR 2014) 15
5.5 Improved tactics for integer and enumerated inequalities (SEPR 2017) 15
5.6 Simplifier uses Examiner-generated implication rules (SEPR 2044) 15
5.7 SPARKSimp 16

6 SPADE Proof Checker 17

7 POGS 18
7.1 More robust treatment of empty VCG file (SEPR 2050) 18

8 SPARKFormat 19
8.1 Formatting other annotations (SEPR 1955) 19
8.2 Reformatting with '*' (SEPR 1975) 19
8.3 Processing of identifiers starting with '$' in annotations (SEPR 2013) 19
8.4 Addition of /properties_indent switch (SEPR 2078) 19

9 SPARKMake 20
9.1 Column alignment in SPARKMake (SEPR 1953) 20
9.2 Order normalization in metafile output (SEPR 1942) 20
9.3 Child package with'ed by subunit not included in metafile (SEPR 1999) 20
9.4 Relative pathnames in metafile and index file output (SEPR 2074) 20

10 Backward incompatibilities 21

11 User manual change summary 22
11.1 Examiner User Manual 22
11.2 SPARK95 – the SPADE Ada95 Kernel 22
11.3 Simplifier User Manual 22
11.4 Generation of RTCs for SPARK Programs User Manual 23
11.5 SPARK95_IO - Input/Output for SPARK95 Programs User Manual 23
11.6 The SPARK Ravenscar Profile User Manual 23
11.7 SPARKMake User Manual 23
11.8 SPARKFormat User Manual 23
11.9 Installation Manuals – NT and Sun 23

12 Limitations and known errors 24
12.1 Tool limitations 24
12.2 Known error summary 26

13 Change Summary from Release 2.0 28
13.1 Release 2.0 - November 1995 28
13.2 Release 2.1 - July 1996 28
13.3 Release 2.5 - March 1997 28
13.4 Release 3.0 - September 1997 28
13.5 Release 4.0 - December 1998 29
13.6 Release 5.0 - June 2000 29
13.7 Release 6.0 - November 2001 30
13.8 Release 6.1 - June 2002 30
13.9 Release 6.3 – December 2002 32
13.10 Release 7.0 – July 2003 32
13.11 Release 7.2 – December 2004 32
13.12 Release 7.3 – February 2006 33
13.13 Release 7.31 – April 2006 34

14 Operating system compatibility 35
14.1 VAX/VMS 35
14.2 SPARC/Solaris 35
14.3 Windows NT, 2000 and XP 35
14.4 Intel/Linux 35
14.5 Apple PowerPC/OS X 35
14.6 Apple Intel/OS X 35

Document Control and References 36
Changes history 36
Changes forecast 36
Document references 37
File under 37

1 Introduction

Continued development of the SPARK Toolset has resulted in the development of a number of useful features that have been incorporated into Toolset Release 7.4.

This document describes changes in the behaviour of all variants of the SPARK Toolset Release 7.4 compared to Release 7.31.

2 Contact Information

For further information about this document please contact Praxis High Integrity Systems:

By phone: +44 (0)1225 823829 (direct line), +44 (0)1225 466991 (exchange)

By FAX: +44 (0)1225 469006

By email: sparkinfo@praxis-his.com

3 SPARK language changes

3.1 Accept Annotation

An accept annotation has been added to the language. Accept annotations can be used to document the acceptance of expected errors and warnings, for example:

```
-- Ineffective assignment to an actual parameter
--# accept Flow_Message, 10, Unused, "X co-ordinate not needed here";
ReadCartesianPair (X => Unused, Y => Height);
--# end accept;
```

The Examiner will suppress the message associated with an accept annotation. An error message will be generated by the Examiner if no error or warning matching the accept annotation exists. The accepted messages are summarised in the Examiner report file.

Details of the annotation and the support provided by the Examiner are given in the SPARK Examiner User Manual section 9.1.

3.2 Obsolete floating-point attributes

In accordance with the Ada95 AARM A5.3 (72.f), obsolete floating-point attributes from Ada 83 are now allowed in SPARK 95 mode. The use of such attributes will generate a Warning 310 "Use of obsolescent Ada 83 language feature." Warning 310 may now be suppressed with a new warning control keyword obsolescent_features.

This change may assist in porting SPARK 83 applications to SPARK 95.

3.3 'Always_Valid assertion

In order to improve the handling of data read from an external variable, the Always_Valid assertion provides a method of indicating to the VC Generator that the values read from a given variable may be assumed to be valid at all times. Previously, and by default, the Examiner has made no assumption about the range or validity of such a value. When a value read from an external variable is known to always lie within a particular range this can be communicated to the VC Generator by use of the Always_Valid assertion, for example:

```
Port : Word;
for Port'Address use ...;
--# assert Port'Always_Valid;
```

Full details of this assertion are given in the Generation of RTCs manual section 4.2.2.

4 Examiner changes

This section documents other significant changes to the SPARK Examiner made for release 7.4. Where appropriate, SPARK Examiner Performance Report (SEPR) numbers are given.

4.1 Index file syntax changes (SEPR 1954)

The Examiner now accepts the token “spec” as an abbreviation for “specification” in index files. See SPARK Examiner User Manual section 4.2.

4.2 Improved syntax error messages for index files (SEPR 1952)

The diagnosis and reporting of syntax errors in index files has been improved. In particular, with the /brief switch selected, the location of the error in the index file is produced in the standard format used by many development enviro nments for error reporting, i.e.:
<filename>:<line_number>:<column_number>.

4.3 Improvements in VC generation (SEPRs 1968, 1979, 1993)

Improvements to the VC generator when tagged types are used (SEPR 1968) and in the VCs generated for checking the right-hand side of modular exponentiation (SEPR 1979). Finally, proof rules regarding the “range” of a Boolean component of a composite constant are no longer generated since Boolean is not ordered in FDL (SEPR 1993).

4.4 External variables in loop bounds (SEPR 1943)

The Examiner now correctly reports an error when an external variable is used in a range constraint controlling a loop rather than failing.

4.5 Improved diagnosis and reporting of syntax errors (SEPR 1967)

Improved diagnosis and reporting of a number of commonly encountered syntax errors in the following areas:

- Annotation following an “is” in a package declaration
- Misplaced own variable clause
- Missing “is” and constituent list in a refinement definition
- Erroneous semicolon preceding the “is” in a subprogram body

- Accidental use of a zero instead of a capital “O” at the start of an identifier
- Unnecessary global or derives annotations on the body of a subprogram, repeating what was defined on the specification.

4.6 Ada 2005 reserved words warning (SEPR 1987)

When analysing SPARK 95 programs, the Examiner will give a new suppressible semantic warning (7) message when one of the new Ada 2005 reserved words are used (overriding, interface, synchronized). The warning may be suppressed using the warning control keyword “ada2005_reserved_words”.

4.7 Ineffective statement in record assignment (SEPR 1988)

When using records with both scalar and array components an incorrect ineffective statement flow error could be reported when assigning to the scalar component. This issue has been resolved.

4.8 Conditional flow errors are now reported as errors (SEPR 1992)

Conditional flow errors are now reported as flow errors, rather than warnings, since they can be the cause of subtle bugs, and should never be ignored or suppressed. The messages that are affected are those with error numbers 501, 504, 601, 602, 605, and 606.

For example, the text of message 602 has changed from:

```
???   Warning    :602: The undefined initial value of X may be used
in the derivation of Y.
```

to:

```
???   Flow Error :602: The undefined initial value of X may be used
in the derivation of Y.
```

4.9 Unconditional flow errors are now reported more clearly (SEPR 1992)

The wording of the error message generated by unconditional flow errors has been changed to remove the confusion between undefined values and undefined variables. The messages that are affected are those with error numbers 20 and 23.

For example, the text of message 23 has changed from:

```
!!!  Flow Error : 23: Statement contains reference(s) to undefined
variable X.
```

to:

```
!!!  Flow Error : 23: Statement contains reference(s) to variable X
which has an undefined value.
```

4.10 Disable FlexLM message box pop-up on Windows (SEPR 2000)

This change only affects Windows.

If the LM_LICENSE_FILE environment variable was not set correctly, running the examiner would cause a pop-up window to appear, asking for the location of the license file, and the result would be saved in the registry. This was confusing and could cause inconsistencies between the registry and the LM_LICENSE_FILE environment variable. With this change, the examiner will fail immediately with an error message if the license file cannot be found.

4.11 Examiner table sizes and limits (SEPR 1997)

The limits of the "megaspark" Examiner have been increased to deal with the largest known SPARK applications.

4.12 FDL Modelling of null string literal "" (SEPR 1176)

In VCs, the null string literal "" is now modelled as an FDL deferred constant called null_string.

4.13 VC Generation of formal unconstrained array parameters (SEPR 1084)

VC Generation for unconstrained array parameters is now correct in all cases, including multi-dimensional arrays and the predefined type String. For each such array parameter X, implicit FDL constants and rules are generated to model X'First, X'Last and so on.

As a result, semantic warning 406 is no longer generated.

Release 2.30 of the Simplifier has also been improved to deal with the common instances of integer and enumerated inequalities that arise from use of such array parameters.

4.14 Optional mangling of FDL reserved words in proof files output (SEPR 2041)

Rather than the previous binary behaviour of the /fdl_identifiers option, it is now possible to pass a string as its value. Passing 'reject' reproduces the behaviour of /fdl and remains the default while a value of 'accept' is equivalent to /nofdl. Any other string is used to prefix any FDL identifiers on output to proof files (i.e. .fdl, .vcg and .rls). This allows the use of programs containing these identifiers with the /exp family of command line options with the Examiner and aids analysis and proof of legacy programs. The old binary usage is still available, but deprecated.

4.15 Examiner exit status (SEPR 2046)

The Examiner now sets its exit status according to the success (or otherwise) of its operation. The value is set according to the table below:

0	Success with no unjustified errors or warnings
1	Unjustified Warnings
2	Unjustified Flow Errors
3	Syntax/Semantic Errors
4-7	Reserved
8	Invocation Error e.g. contradictory command-line switches
9	Internal Error e.g. table overflow or internal exception

4.16 System.Bit_Order available in configuration file (SEPR 2002)

The type System.Bit_Order is now implicitly declared in the configuration file, along with the appropriate enumeration literals (Low_Order_First and High_Order_First) and System.Default_Bit_Order as a deferred constant. It is possible to override the default with a declaration within package System in the configuration file such as:

```
Default_Bit_Order : constant Bit_Order := Low_Order_First;
```

4.17 XML and HTML output of justifications (SEPR 1980)

XML and HTML output of justifications has been introduced, in line with the standard text output. The human readability of generated XML has also been improved. The XML Scheme Definition file (sparkreport.xsd) has been updated to describe the structure of the justification summary.

4.18 Examiner no longer accepts out of range literals in modular expressions (SEPR 2057)

Previously the Examiner could accept an out of range literal in a modular expression if the literal appeared beneath a relational operator. A range check is now performed to detect static errors of this type that would previously have been caught at compile time. For example:

```
type T is mod 128;
B : T := 10;
A : Boolean;

A := (B /= 128);
```

is illegal as 128 is outside the valid range of type T (0..127). Checks of this kind are also performed on signed integer types if a base type assertion has been provided.

4.19 SPARK95_IO interface change (SEPR 2065)

The `File` formal parameter to the `Close` and `Delete` procedures is now of mode **`in out`** to allow for deallocation of the underlying `Text_IO.File_Type` object.

4.20 'Size not static for non scalar types (SEPR 2051)

'Size is only considered to be static where the prefix denotes a static scalar (sub-)type. The Examiner now correctly enforces this rule.

4.21 Metafile handling improved (SEPR 2071)

Previously the usage of nested metafiles was limited to full pathnames. It is now possible to use relative pathnames to reference other metafiles from within a metafile.

4.22 Ada.Real_Time package definition changed (SEPR 2081)

The definition of Ada.Real_Time has changed such that the own variable ClockTime has a type announcement of the private type Time:

```
--# own protected in ClockTime : Time;
```

4.23 Detection of illegal constrained array subtype declarations (SEPR 2084)

The Examiner now correctly rejects a constrained array subtype declaration where the constraining index subtype is not compatible with the range of the designated unconstrained array type. For instance, the following declaration for ST is now rejected:

```
type T is array (Positive range <>) of Integer;
subtype I  is Natural range 0 .. 15;
subtype ST is T (I); -- Illegal since I'First < Positive'First
```

Violations of this rule are reported with semantic error 402: “Constraint error will be raised here.”

Note that a compiler is also obliged reject such a declaration.

5 SPADE Simplifier

Simplifier 2.30 ships with toolset release 7.4. This incorporates the following improvements and features:

5.1 User defined proof rules (SEPR 1969 & 1970)

The simplifier could fail with an "infer/3 undefined" message when processing user-defined rules. This fault has been rectified (SEPR 1969). An error in pattern matching in user-defined rules has also been corrected (SEPR 1970).

5.2 Standardisation of behaviour of proof rules (SEPR 1974)

The simplifier could behave differently for the same rule defined in the SPARK rule file (RLS) and the user-defined rule file (RLU). This anomaly has been rectified.

5.3 New /noexpression_reduction option (SEPR 1986)

A new switch, /noexpression_reduction inhibits the simplifier from performing expression reduction (described in the SPADE Automatic Simplifier User Manual section 3.10). The use of the switch is described in section 5 of the Simplifier User Manual.

5.4 Earlier use of ground user-defined replacement rules (SEPR 2014)

If an user-defined rule, defined in a RLU file, gives a "may_be_replaced_by" rule that contains no variables, has no side-conditions, and the left hand side is the name of an FDL constant, then such rules are applied earlier in the simplification process. This improves the performance of latter simplification tactics in some cases.

5.5 Improved tactics for integer and enumerated inequalities (SEPR 2017)

A new family of proof tactics have been added for enumerated and integer-typed inequalities where the proof requires a proof by transitivity. In particular, these tactics are able to prove the VCs that arise from the common cases of a SPARK "for" loop which iterates over the elements of an unconstrained array parameter, taking advantage of the improved VC generation for such objects described above.

5.6 Simplifier uses Examiner-generated implication rules (SEPR 2044)

Version 2.30 of the Simplifier has the ability to handle user-defined non-ground implication rules. Examiner-generated implication rules are now also used by the Simplifier.

5.7 SPARKSimp

SPARKSimp 3.1 ships with toolset release 7.4 and incorporates the following new features.

5.7.1 New /x=Simpexec option (SEPR 1985)

SPARKSimp has a new switch /x=Simpexec which allows an alternative simplifier executable to be used. Details are given the SPADE Automatic Simplifier User Manual section 8.

5.7.2 Implementation of /p=N (multi-processing) switch improved (SEPR 1945)

The implementation of the /p=N switch has been substantially improved. In particular, SPARKSimp is now able to make use of all the CPU time available on multi-processor and multi-core machines. On such machines, simplification times will be improved owing to better utilization of the available processors.

6 SPADE Proof Checker

SPADE Proof Checker 2.08 ships with toolset release 7.4. This incorporates a single correction to prevent the rejection of predefined PROLOG operators (such as “block”) where there may be a conflict with the use of the same identifiers in FDL.

There is no other user-visible difference in behaviour between releases 2.07 and 2.08.

7 POGS

POGS 4.6 ships with toolset release 7.4 and incorporates the following new features.

7.1 More robust treatment of empty VCG file (SEPR 2050)

Previously an empty VCG file would cause POGS to loop indefinitely while searching for the subprogram declaration. This case is now detected and an appropriate error message issued.

8 SPARKFormat

SPARKFormat 7.4 ships with toolset release 7.4. This incorporates the following improvements and features:

8.1 Formatting other annotations (SEPR 1955)

SPARKFormat now supports formatting of the own, initializes and inherit annotations, and has three new switches for controlling the handling of these cases.

8.2 Reformatting with '*' (SEPR 1975)

A problem with SPARKFormat where the '*' character was not necessarily the first item in an import list has been resolved.

8.3 Processing of identifiers starting with '$' in annotations (SEPR 2013)

SPARKFormat 7.4 allows identifiers in annoations to start with a '$' character. This may be useful in conjunction with macro-processing tools such as GNATPREP.

8.4 Addition of /properties_indent switch (SEPR 2078)

SPARKFormat now supports control of the indentation of property lists of own variables such that they may be placed inline or indented a set number of characters.

9 SPARKMake

SPARKMake 7.4 ships with toolset release 7.4 and incorporates the following new features.

9.1 Column alignment in SPARKMake (SEPR 1953)

The readability of metafiles produced by SPARKMake has been improved by correctly aligning columns that describe a specification location with those that describe a body location.

9.2 Order normalization in metafile output (SEPR 1942)

Ordering of files output to metafiles created by SPARKMake no longer differs on Windows and Solaris.

9.3 Child package with'ed by subunit not included in metafile (SEPR 1999)

Previously a child package with'ed by a separate of the parent package would not always be included in the metafile, causing analysis to fail. This behaviour is now corrected.

9.4 Relative pathnames in metafile and index file output (SEPR 2074)

SPARKMake is capable of outputting relative pathnames to metafiles and index files. This behaviour is selectable using the new switch /path, which can be set equal to either full or absolute. The default is full to preserve existing behaviour.

10 Backward incompatibilities

There are no known backward incompatibilities introduced between Examiner Releases 7.31 and 7.4.

Note however, that the exact wording of the messages generated by conditional and unconditional flow errors has changed, and that this might cause changes in the output generated by the examiner. (See sections 4.8 and 4.9 for further details).

Secondly, semantic warning 406 is no longer generated following SEPR 1084. If regression analysis shows that this warning was previously generated, then VCs for the offending subprograms should be re-generated and re-simplified.

11 User manual change summary

The following user manuals have been changed between Examiner Releases 7.31 and 7.4.

- SPARK Examiner User Manual
- SPARK95 – The SPADE Ada95 Kernel
- SPADE Automatic Simplifier User Manual
- Generation of RTCs for SPARK Programs User Manual
- SPARK95_IO - Input/Output for SPARK95 Programs User Manual
- The SPARK Ravenscar Profile User Manual
- SPARKMake User Manual
- SPARKFormat User Manual
- Installation Manuals – NT and Sun

11.1 Examiner User Manual

- Documentation of accept annotations.
- Description of warning 169 updated to include the name of the directly updated own variable.
- Documentation of “spec” as an abbreviation for “specification” in index files.
- Alteration of command line option for handling FDL reserved words.
- Documentation of exit status values.
- Documentation of System.Bit_Order in configuration file.

11.2 SPARK95 – the SPADE Ada95 Kernel

- Change to allow obsolete SPARK83 floating-point attributes in SPARK95 mode.

11.3 Simplifier User Manual

- Documentation of SPARKSimp /x=Simpexec option.

- Documentation of /noexpression_reduction option.

11.4 Generation of RTCs for SPARK Programs User Manual

- Documentation of the Always_Valid assertion.

11.5 SPARK95_IO - Input/Output for SPARK95 Programs User Manual

- Documentation of the interface change to the Close and Delete subprograms.

11.6 The SPARK Ravenscar Profile User Manual

- Alteration to the definition of the Ada.Real_Time package.

11.7 SPARKMake User Manual

- Documented /path option.

11.8 SPARKFormat User Manual

- Documented formatting of inherit, initialization and own variable clauses.
- Documented addition of new /properties_indent option.

11.9 Installation Manuals – NT and Sun

- Updates to installation procedures for release 7.4.

12 Limitations and known errors

12.1 Tool limitations

This section describes limitations of the Examiner tool arising mainly from incomplete implementation of planned features. Where appropriate a SPARK Examiner Performance Report (SEPR) number is given.

12.1.1 General

1 The SPARK 95 language definition removes the distinction between initial and later declarative items; this distinction remains in force in the Examiner that requires SPARK 83 declaration orders even in SPARK 95 mode. (SEPR 813)

2 The Examiner does not yet permit the use of 8-bit characters in SPARK 95 user-defined identifiers. (SEPR 818)

3 Universal expressions in a modular context may sometimes require type qualification. (SEPR 1591)

4 The Examiner does not yet permit the use of "use type" following an embedded package specification. (SEPR 747)

5 The Examiner does not yet permit the renaming of packages in the same way that subprograms can be renamed. (SEPR 1391)

6 The Examiner does not yet allow the 'Base attribute when not used as a prefix. (SEPR 1114)

7 The Examiner does not yet allow S'Range where S is scalar. (SEPR 1115)

12.1.2 Verification Condition Generation and Run-time Checks

1 Ada string inequality is not modelled. (SEPR 712)

2 VCs involving string catenation that includes the character ASCII.NUL will be incorrect. (SEPR 661)

3 Aggregates of multi-dimensional arrays cannot be modelled although aggregates of arrays of arrays can. (SEPR 590)

4 Verification conditions involving real numbers are evaluated using infinite precision or perfect arithmetic; this allows the correctness of an algorithm to be shown but cannot guard against the effects of cumulative rounding errors for example.

5 The Examiner does not generate VCs for package initialization parts. Statically determinable constraint errors will be detected during well-formation checking of package initialization. (SEPR 288)

6 The VC Generator cannot model the implementation-dependent attributes of floating and fixed-point types; see section 12.1.3.

7 The Examiner does not generate hypotheses for a task discriminant if the discriminant is referenced in the task body. This may result in an RTC VC that the Simplifier cannot prove. (SEPR 2088)

12.1.3 Attribute limitations

12.1.3.1 Unimplemented attributes

The following attributes are officially supported by SPARK according to the language definition, but are not yet implemented by the Examiner. The Examiner will generate error number 30 ("Attribute XXX is not yet implemented in the Examiner") if you try to use them.

- Adjacent
- Compose
- Copy_Sign
- Leading_Part
- Remainder
- Scaling
- Exponent
- Fraction
- Machine
- Model
- Rounding
- Truncation
- Unbiased_Rounding

Note that these are all function-like attributes concerning floating- and fixed-point types.

12.1.3.2 Unevaluable attributes

The Machine_* and Model_* attributes are accepted by the Examiner, but it does not know how to statically evaluate them since they are inherently implementation dependent. For example, the package:

```
package F is
   type T is digits 6 range –10.0 .. 10.0;
   C : constant := T'Machine_Emax;
end F;
```

is legal SPARK, but the Examiner does not know the actual numeric value of C.

12.2 Known error summary

This section lists known errors in the Examiner that are awaiting investigation and correction.

1 The SPARK rule that array actual parameters must have the same bounds as the formal parameter is not checked for function parameters where the actual parameter is a subtype of an unconstrained array type. Since subtype bounds are static in SPARK errors of this kind should be detected by an Ada compiler. If not an unconditional run-time error will occur. (SEPR 1060)

2 The Examiner permits the body of a subprogram to be entirely made up of proof statements thus breaching the Ada rule that at least one Ada statement must be present. (SEPR 278)

3 Where a package declares two or more private types the Examiner permits mutual recursion between their definitions in the private part of the package. (SEPR 848)

4 The Examiner does not take due account of a range constraint when determining the subtype of a loop variable; this affects completeness checking of case statements within the loop. For example `for I in Integer range 1..4 loop` would require only values 1, 2, 3 and 4 to be covered by the case statement. (SEPR 693)

5 When summarising the counts of pragmas found during an analysis the totals may depend on whether units are selected via the command line (or metafile) or using the index mechanism. The difference affects only pragmas placed *between* program units and arises because placing a file name on the command line causes the entire *file* to be analysed whereas selecting it using indexes causes only the required *unit* to be read from the file. (SEPR 483)

6 A view conversion of a tagged array element causes the Examiner to fail (SEPR 2052).

For example:

```
T := Parent_Type (Child_Array (I));
```

Causes an Examiner failure. Workaround is to introduce a temporary variable and use two assignment statements, such as:

```
Tmp := Child_Array (I);
T   := Parent_Type (Tmp);
```

7 The Examiner fails to reject the use of a task discriminant in a task body in a context where a static or constant expression is required. For example, if a task type TT has a discriminant X of type I, then the declarations

```
task body TT
is
   type T is range 1 .. X; -- illegal Ada
   C : constant I := X;    -- illegal SPARK
   …
```

are incorrectly accepted by the Examiner. (SEPR 2087)

13 Change Summary from Release 2.0

A release note detailing changes from the previous version accompanies each Examiner Release; this section simply summarises the various changes that have been made.

13.1 Release 2.0 - November 1995

Release 2.0 added:

- static expression evaluation;
- variable initialization at declaration;
- full-range scalar subtypes; and
- operator renaming in package specifications.

13.2 Release 2.1 - July 1996

Release 2.1 added:

- facilities for proof of absence of run-time errors

13.3 Release 2.5 - March 1997

Release 2.5 was distributed with “High Integrity Ada - the SPARK Approach” and provided initial facilities for SPARK 95

13.4 Release 3.0 - September 1997

Windows NT was supported for the first time with this release. Release 3.0 also added:

- additional SPARK 95 support;
- flow analysis of record fields;
- command line meta files;
- named numbers;
- unqualified string literals;
- moded global annotations; and

- optional information flow analysis.

13.5 Release 4.0 - December 1998

With Release 4.0 we upgraded all users to a single product standard supporting SPARK 83, SPARK 95 and analysis options up to an including proof facilities. New features were:

- full implementation of public and private child packages;
- default switch file; and
- provision of the INFORMED design document.

13.6 Release 5.0 - June 2000

- Enhanced proof support:
  I. facilities for proof of programs containing “abstract state”;
  II. addition of quantified expressions;
  III. proof rule generation for enumeration types;
  IV. identification of the kind and source of each VC;
  V. suppression of trivially true VCs;
  VI. Proof Obligation Summariser tool (POGS)
- Optional HTML output files with hyperlinks that can be “browsed” interactively
- Better support for common Ada file naming conventions
- User-selectable annotation character
- Improved suppression of analysis were results might otherwise be misleading
- Static expression evaluation in proof contexts
- Singleton enumeration types
- Revised SPARK_IO package
- Error numbering

13.7 Release 6.0 - November 2001

- Introduction of “external variables” to simplify modelling of the interactions between a SPARK program and its external environment.
- Addition of the “null derives” annotation to describe information flows which affect only the external environment.
- Introduction of modular types
- Use of loop labels in exit statements
- Use of global modes on function subprograms
- Extended support for predefined types such as Long_Integer
- Simplified run-time check generation for own variables
- Relaxation of need for mandatory type announcement of own variables
- Plain output option to simplify comparisons of Examiner output files
- Platform-independent switch files and metafiles
- Support for intentionally infinite loops
- Detection of own variables that can never be initialized
- Detection of unusable private types
- Extra refinement checks on global variables when performing data flow analysis
- Detection of unnecessary others clause in case statements
- Extensions to the POGS tool
- Improved error messages to distinguish different cases of variables which are “not declared or visible”
- Improved SPADE Simplifier Release 2.0
- New “SPARKSimp” Tool

13.8 Release 6.1 - June 2002

- Introduction of tagged types

- Introduction of type assertion annotations
- Introduction of modular subtypes
- Introduction of the configuration file
- Introduction of the `help` command line switch
- Demo Examiner now runs on Linux.
- VCG generation for inherited operations of tagged types
- Improved handling of null derives
- Attributes ‘Floor and ‘Ceiling implemented
- Detection of duplicate record fields
- Improved overflow checks on universal integer expressions
- Corrected handling of loop invariants in while loops
- Strengthened behaviour of /noecho option
- Trapping non-positive accuracy in real type declaration
- Recursion in meta-files and index-files
- Improved handling of Address clauses
- Improved handling of Import and Interface pragmas
- VCG Modelling of Boolean membership operators
- Simplification of common Integer inequalities
- Simplification of common enumerated inequalities
- Simplification of VCs involving quantified expressions
- Simplifier performance
- Checker has new built-in rule families: MODULAR, BITWISE and ENUMERATION
- Proved by review option in POGS

13.9 Release 6.3 – December 2002

Release 6.3 of the toolset was constructed to accompany the textbook "High Integrity Software: The SPARK Approach to Safety and Security" by John Barnes, but was not delivered to other users. Its main new features were:

- Slight revision to the rules regarding the placement of tagged type declarations.
- Correction to the modelling of Boolean type membership operators in the verification conditions.
- Support for generating VCs that allow the verification of the Liskov Subsitution Principal (LSP) for tagged types and their operations.
- Dramatically improved performance of the Simplifier, particularly in the simplification of quantified expressions.

13.10 Release 7.0 – July 2003

Release 7.0 of the toolset comprised:

Examiner version 7.0

Simplifier version 2.12

Release 7.0 added:

- Ravenscar profile extensions to the language.
- Support for Ada.Interrupts and Ada.Real_Time in the configuration file.
- The new /noduration command line switch.
- VC generation for unconstrained formal parameters.
- Suppression of VC generation for illegal function bodies.
- New "SPARKFormat" tool.

13.11 Release 7.2 – December 2004

Release 7.2 of the toolset comprised:

Examiner version 7.2

Simplifier version 2.17

Release 7.2 added:

- Unconstrained string constants to the language.
- Instantiation of Unchecked_Conversion to the language.
- (Full) record subtypes to the language.
- Declaration of subprograms in the private part of packages.
- Refined proof annotations for private types.
- % suffix for referring to value of variable on entry to loop in proof contexts.
- Extra hypotheses for local variables.
- Suppression of VC generation for illegal function bodies.
- Replacement rules for composite constants.
- Concurrent simplification with SPARKSimp.
- Improved simplification of VCs with large structured objects.
- Improved simplification of arithmetic and logical expressions.
- New "SPARKMake" tool.

13.12 Release 7.3 – February 2006

Release 7.3 of the toolset comprised:

Examiner version 7.3

Simplifier 2.22

Checker 2.06

Significant features of this release included:

- Improved VC Generation.
- New "error explanations" switch for Examiner.
- Generation of proof rules for 'Size.
- Improved diagnosis and reporting of common syntax errors.

- Error and warning count summary in Examiner output.
- Use of pragma Import to complete an external own variable.
- Correction to FDL declaration order for private and announced types.
- New Simplifier tactics for dealing with rational inequalities and Examiner-generated proof rules.
- User-defined proof rules for the Simplifier.
- New Checker rules for MODULAR expressions.
- Significant performance improvement for Simplifier and Checker.

13.13 Release 7.31 – April 2006

Release 7.31 of the toolset comprised:

Examiner version 7.31

Simplifier 2.24

Checker 2.07

Significant features of this release included:

- Correction to flow analysis of array element “out” parameters.
- Port of “Demo” toolset to Mac OS X.
- New /typecheck option in Simplifier.
- Improved validity checking of user-defined proof rules in the Simplifier.
- Correction to ARITH proof rules in the Checker.

14 Operating system compatibility

14.1 VAX/VMS

Toolset release 7.4 is not supported on VAX/VMS. If you are currently using toolset release 7.31 or earlier on VAX/VMS, please contact SPARK team to discuss availability on other platforms.

14.2 SPARC/Solaris

The toolset is compatible with Solaris 5.6 through to 10 including those with a 64-bit kernel.

14.3 Windows NT, 2000 and XP

The toolset is compatible with Windows NT 4.0, Windows 2000, and Windows XP. The executables are also known to work on Windows 95 and 98; however, use of the toolset on these operating systems is unsupported. The FlexLM licence manager software only runs on Windows NT, Windows 2000, or Windows XP.

14.4 Intel/Linux

All the toolset, with the exception of the Checker, is compatible with Intel-based Linux operating systems. Only the "FreeDemo" version of the toolset is currently available for Linux to support buyers of John Barnes' "SPARK Book." If you require a full professional SPARK toolset for Linux, then please contact us.

14.5 Apple PowerPC/OS X

All the toolset, with the exception of the Checker, is compatible with Apple's PowerPC OS X/Darwin operating systems. Only the "FreeDemo" version of the toolset is currently available for OS X to support buyers of John Barnes' "SPARK Book." If you require a full professional SPARK toolset for OS X, then please contact us.

14.6 Apple Intel/OS X

The PowerPC/OS X binaries should run unmodified on Intel/OS X under Apple's "Rosetta" system. A fully native version of the toolset for Intel/OS X may be available in future.

Document Control and References

Praxis High Integrity Systems Limited, 20 Manvers Street, Bath BA1 1PX, UK.
Copyright © Praxis High Integrity Systems Limited 2006. All rights reserved.

Changes history

Issue 0.1 (21st June 2006): First draft for release 7.31d01.

Issue 0.2 (23rd June 2006): Added section regarding SEPR 1988

Issue 0.3 (3rd July 2006): Increased Examiner table sizes for release 7.31d01.

Issue 0.4 (12th July 2006): New sections for release 7.31d02.

Issue 0.5 (3rd September 2006): Update for release 7.31d03.

Issue 0.6 (7th September 2006): Updated for release 7.31d04.

Issue 0.7 (21st November 2006): Updated for release 7.31d05.

Issue 0.8 (24th November 2006): Updated for release 7.31d06.

Issue 0.9 (28th November 2006): Updated for release 7.31d07.

Issue 1.0 (6th December 2006): Updated for release 7.31d10.

Issue 1.1 (11th December 2006): Updated for change to SPARKMake /path documentation.

Issue 1.2 (18th December 2006): Update for releases 7.31d09 to d10.

Issue 1.3 (19th December 2006): Updates for Examiner 7.4, Simplifier 2.30 and Checker 2.08.

Issue 1.4 (20th December 2006): Updated known limitations and errors for release 7.4.

Issue 1.5 (20th December 2006): Definitive issue for release 7.4 following review S.P0468.79.92.

Changes forecast

Updates following review.

Document references

File under

CVSROOT/userdocs/Examiner_RN_7p4.doc